AMENDMENTS TO THE CLAIMS

1. (currently amended) In a computer system, a method for implementing and using a filter object ~~which generates an output in response to an input of the filter object, wherein the output of the filter object depends on the input and a state of the filter object, wherein the state of the filter object includes a minimum amount of information necessary to determine the output of the filter object~~, the method comprising:

providing the filter object, the filter object including a state, and the filter object being represented by:

an output equations ~~performed to~~for generating~~e~~ ~~the~~ an output ~~in response to the input~~ of the filter object, and

a state equation for updating the state of the filter object~~, the equations including the state of the filter object~~; ~~and~~

implementing the filter object; and

using the filter object in a first dynamically typed text-based programming environment, the using the filter object including:

receiving an input at the filter object,

identifying a first state of the filter object,

processing the output equation to determine an output of the filter object based on the input of the filter object and the first state of the filter object,

processing the state equation to determine a second state of the filter object based on the input of the filter object and the first state of the filter object,

retaining the second state of the filter object in the first dynamically typed text-based programming environment, and

making the second state available after the output equation of the filter object is processed~~;~~

~~wherein the filter object is implemented and used in a first dynamically typed text-based programming environment, wherein the output of the filter object is determined based on a present input of the filter object and a present state of the filter object, and wherein the state of the filter object contains information about a previous input of the filter object.~~

2. (currently amended) The method of claim 1 wherein the filter object retains a final ~~value of the~~ state obtained ~~as a result of~~after processing the input of the filter object.

3. (currently amended) The method of claim 2 wherein the final ~~value of the~~ state retained in the filter object is used as an initial ~~value of the~~ state for processing ~~the~~ a second input of the ~~system~~filter object.

4. (currently amended) The method of claim 1 further comprising the step of resetting ~~the~~ a state of the filter object retained in the filter object.

5. (currently amended) The method of claim 1 further comprising the step of presetting ~~the~~ a state of the filter object retained in the filter object.

6. (canceled)

7. (canceled)

8. (currently amended) The method of claim 1 wherein the filter object ~~is utilized to generate~~generates code ~~to~~ for implementing a corresponding filter algorithm ~~separate from the filter object implementation~~.

9. (currently amended) The method of claim 1 wherein the filter object ~~is utilized to generate~~generates code to implement a ~~corresponding~~ test bench or a filter analysis.

10. (currently amended) The method of claim 8 wherein the filter object executes in a simulation environment, and the generated code is ~~can be~~ executed~~, directly or via a suitable compilation process, on the host machine, but~~ outside ~~the~~ a context of the~~a~~ simulation environment in which the filter object executes.

11. (currently amended) The method of claim 10 wherein the generated code is in a textual language.

12. (currently amended) The method of claim 10 wherein the generated code is in a graphical description language.

13. (currently amended) The method of claim 8 wherein the filter object executes in a simulation environment, and the generated code ~~can be~~is executed~~, directly or via a suitable compilation process, on the host machine,~~ within a~~the~~ context of the~~a~~ simulation environment ~~on~~ in which the filter object executes.

14. (currently amended) The method of claim 8 wherein the generated ~~filter~~ code ~~can be~~is executed~~, directly or via a suitable compilation process,~~ in an environment separate from ~~the~~ a computer system used for a simulation of the filter object, including an embedded system implementation.

15. (currently amended) The method of claim 14 wherein the generated code is in a textual language.

16. (currently amended) The method of claim 14 wherein the generated code is in a graphical description language.

17. (currently amended) The method of claim 14 wherein the generated code is suitable for use with a software implementation, the software implementation being adapted for ~~including~~ use of the generated code on at least one of a general purpose processor, a digital signal processor, and a programmable computer architecture.

18. (currently amended) The method of claim 14 where the generated code is suitable for use with a hardware implementation, the hardware implementation including use with at least one of a Field Programmable Gate Array (FPGA), Complex Programmable Logic Device (CPLD), and Application Specific Integrated Circuit (ASIC) device, the generated code being written in hardware description language.

19. (currently amended) The method of claim 8 wherein the code is written in a high-level programming language.

20. (currently amended) The method of claim 8 wherein the code is written in a low-level machine or assembly language.

21. (currently amended) A~~In a~~ computer-implemented~~ system, a~~ method for generating an output of ~~the~~ a system in response to an input to~~of~~ the system, the method comprising~~ the steps of~~:

implementing the system using a dynamically typed text-based programming environment; and

using the system in the dynamically typed text-based programming environment, the system:

determining an output of the system based on an input to the system and a first state of the system;

determining a second state of the system based on the input to the system and the first state of the system; and

retaining the second state of the system in the dynamically typed text-based programming environment so that the second state is available after the output of the system is determined~~specifying a state of the system that includes a minimum amount of information that is necessary to determine the output of the system;~~

~~retaining the state of the system in a memory;~~

~~providing to the system the state of the system retained in the memory; and~~

~~determining the output of the system depending on the input and a state of the system;~~

~~wherein the method is implemented in a dynamically typed text-based programming environment, wherein the output of the system determined based on a present input of the system and a present state of the system, and wherein the state of the system contains information about a previous input of the system~~.

22. (currently amended) The method of claim 21 further comprising ~~the step of~~ specifying equations that the system ~~performs~~ processes to generate the output of the system ~~from~~ using the input to the system and ~~the~~ a state of the system.

23. (currently amended) The method of claim 21 further comprising ~~the step of~~ controlling ~~the~~ a state of the system retained in the memory.

24. (currently amended) The method of claim 23 wherein ~~the~~ a state of the system retained in the memory is reset to provide a zero initial state to the system.

25. (currently amended) The method of claim 23 wherein ~~the~~ a state of the system retained in the memory is set to a particular value entered by a user.

26. (currently amended) The method of claim 21 wherein ~~the~~ a state of the system retained in the memory includes a final ~~value of the~~ state obtained ~~as a result of~~by processing the input ~~of~~ to the system.

27. (currently amended) The method of claim 21 wherein ~~the~~ a state of the system ~~provided to the system includes~~ is an initial state of the system for processing the input of the system.

28. (currently amended) A computer readable medium holding instructions executable in a computer that provides a dynamically typed text-based programming environment~~, wherein the computer generates an output of an object in response to an input of the object~~, the instructions comprising:

providing an ~~class~~object, the object being an instance of ~~the~~ a class;

~~specifying a state of the object that includes a minimum amount of information that is necessary to determine the output of the system, the state being a property of the object~~ ~~;~~

determining an output of the object based on an input to the object and a first state of the object;

determining a second state of the object based on the input to the object and the first state of the object; ~~and~~

retaining the second state of the object in the dynamically typed text-based programming environment; and

making the second state available after determining the output of the object~~;~~

~~determining the output of the object depending on the input and the state of the system;~~

~~wherein the output of the object is determined based on a present input of the object and a present state of the object, and wherein the state of the object contains information about a previous input of the object.~~

29. (original) The medium of claim 28 further comprising the step of instantiating the object from the class.

30. (original) The medium of claim 28 wherein the object includes an adaptive filter object.

31. (currently amended) The medium of claim 30 wherein the adaptive filter object includes an adapting algorithm that the adaptive filter ~~performs~~implements.

32. (original) The medium of claim 28 wherein the object includes a discrete time filter object.

33. (currently amended) The medium of claim 28 further comprising ~~the step of~~ controlling properties of the object including ~~the~~ a state of the object.

34. (original) The medium of claim 33 wherein the state of the object is reset to zero.

35. (currently amended) The medium of claim 28 further comprising ~~the step of~~ inheriting a state property corresponding to ~~the~~ a state of the object from an abstract class.

36. (currently amended) The medium of claim 28 further comprising ~~the step of~~ providing the class with methods which operate on the object of the class.

37. (currently amended) A system for implementing a filter object~~ which generates an output in response to an input of the filter object, wherein the output of the filter object depends on the input and a state of the filter object, wherein the state of the filter object includes a minimum amount of information necessary to determine the output of the filter object~~, the ~~method~~ system comprising:

a processor configured to process:

an output equation of the filter object to determine an output of the filter object based on an input to the filter object and a first state of the filter object;

a state equation of the filter object to determine a second state of the filter object based on the input to the filter object and the first state of the filter object; and

a memory for retaining the second state of the filter object in a dynamically typed text-based programming environment so that the second state is available after the output equation is processed~~; and~~

~~a state equation processing unit for generating a new state of the filter object based on the state of the filter object retained in the memory and the input of the filter object;~~

~~wherein the filter object is implemented and used in a first dynamically typed text-based programming environment, wherein the output of the filter object is determined based on a present input of the filter object and a present state of the filter object, and wherein the state of the filter object contains information about a previous input of the filter object~~.

38. (canceled)

39. (currently amended) The system of claim ~~38~~37, wherein the ~~new~~ second state retained in the memory is used as a state of the filter object in processing a next input of the filter object.

40. (canceled)

41. (canceled)

42. (currently amended) The system of claim 37, wherein ~~the~~ a state of the filter object retained in the memory is reset to provide a zero initial state.

43. (currently amended) The system of claim 37, wherein ~~the~~ a state of the filter object retained in the memory is set to a particular value entered by a user.

44. (previously presented) The method of claim 1, wherein the filter object operates on a sample-by-sample, block-by-block or frame-by-frame basis.

45. (previously presented) The method of claim 21, wherein the system operates on a sample-by-sample, block-by-block or frame-by-frame basis.

46. (previously presented) The medium of claim 28, wherein the object operates on a sample-by-sample, block-by-block or frame-by-frame basis.

47. (previously presented) The system of claim 37, wherein the input of the filter object comprises a single piece of sample data, a sequence of sample data or multiple sequences of sample data.